

This listing of claims will replace all prior versions, and listings, of claims in the application.

**LISTING OF CLAIMS:**

Claim 1. (Canceled)

Claim 2. (Currently Amended) The multistage microprocessor pipeline structure of claim [[1]] 19, wherein the width determination logic means uses data about the length of operands stored in ~~a register file tags module~~ said register value information means that stores value bit information about each operand in the register file, including the sign and width of each operand, and one or more leading bits of one or more bytes of each operand, which are ~~examined for data overflow from a lesser significant slice to a more significant slice~~ used to determine the number and which slices to enable for execution.

Claim 3. (Original) The multistage microprocessor pipeline structure of claim 2, wherein the value bit information includes a sign of an operand in one bit, a register data width in bytes of the operand value in two bits, and one or more leading bits of one or more of the most significant bytes of the operand.

Claim 4. (Original) The multistage microprocessor pipeline structure of claim 2, wherein the output of the decoder indicates two source registers and one destination register, and an instruction operation code, ~~and the register file tags module~~ said register value information means and the instruction operation code are used to determine the number of slices required for executing the corresponding processing instruction, and those number of slices are enabled in subsequent cycles in the pipeline structure.

Claim 5. (Currently Amended) The multistage microprocessor pipeline structure of claim [[1]] 19, wherein a cache tag file stores addresses of the cache memory to write to

and read from, and a width tag file stores the width of data stored in each memory address.

Claim 6. (Currently Amended) The multistage microprocessor pipeline structure of claim [[1]] 19, wherein the width determination ~~logic~~ means outputs the width control bits to enable data flow and computation in the slices.

Claim 7. (Currently Amended) The multistage microprocessor pipeline structure of claim [[1]] 19, wherein enabling and disabling of each slice is accomplished by clock gating, where during enabling, clock signals allow data to proceed into and through a slice, and during disabling, clock signals block the flow of data into and through a slice.

Claim 8. (Currently Amended) The multistage microprocessor pipeline structure of claim 2, wherein the width determination ~~logic~~ means determines the likelihood of a data overflow being generated from a narrow slice operation by examining one or more leading bits of the operands which are stored in ~~the register file tags module~~ said register value information means and generates one of three determinations:

no data overflow is guaranteed, and the effective operation width is determined by the width of the narrow operands;

data overflow is guaranteed, and the effective operation width must be one byte larger than the width of the narrow operands;

data overflow is possible but not certain, wherein a carry into the bits examined is propagated as a carry out.

Claim 9. (Currently Amended) The multistage microprocessor pipeline structure of claim [[1]] 19, wherein following execution and completion of a processing operation by the arithmetic logic unit, the width of the value of the ~~processing~~ processed operation result is determined by the width determination means, after which ~~the width determination logic determines value bit~~ a register information value for the processing operation result is formed by combining its sign bit, value width and one or more leading bits for its one or more leading bytes, said register information value being written to

said register value information means and said operation result is which is then written to a destination register in the register file.

Claim 10. (Previously Presented) A method for reducing logic activity in the execution of an operation in a processor comprising the steps of:

- selecting at least one operand associated with said operation,
- looking up a width and a value of selected bits of said at least one operand,
- determining a prediction of arithmetic overflow, based upon the width and the value of said selected bits of said at least one operand,
- determining an effective width of said operation based upon the width of said at least one operand, a function specified by said operation, and said prediction of arithmetic overflow,
- enabling the width of the resources in said processor corresponding to said effective width of said operation for executing said operation,
- executing said operation over the enabled width of the resources,
- determining the width of the result of said operation based upon the step of executing.

Claim 11. (Original) The method of claim 10, including saving the width of the result of said operation, and saving said result of said operation.

Claim 12. (Original) The method of claim 10, wherein said step of looking up a width and a value of selected bits includes dedicated hardware for holding and retrieving said width and said value of selected bits.

Claim 13. (Original) The method of claim 10, wherein the processor includes a register file, an arithmetic unit, a memory path, and a cache memory, and the register file, the arithmetic unit, and the cache memory are divided into a plurality of slices, each of which is of a reduced bit granularity, and the bits in all of the slice form a full width word in the processor.

Claim 14. (Original) The method of claim 13, wherein at least one slice is of 8 bit granularity.

Claim 15. (Original) The method of claim 13, wherein at least one slice is of 16 bit granularity.

Claim 16. (Original) The method of claim 13, wherein said step of enabling includes logic to enable a required number of slices to execute the operation.

Claim 17. (Currently Amended) A processor comprising:

a plurality of slices, each of which is a portion of a full width word of the processor, wherein each slice comprises a portion of a register file, a portion of functional units, a portion of a memory path, a portion of a cache memory, and a portion of other resources required to perform operations in the processor,

logic to save and retrieve a width and selected bits of operands used to perform an operation in the processor,

logic to determine a prediction of arithmetic overflow when performing the operation, based upon the width and the selected bits of the operands used to perform the operation,

logic to determine a number of slices required to perform the operation based upon the width of one or more operands, the functionality of the operation, and the prediction of arithmetic overflow, and,

logic to activate the determined number of slices required to perform the operation.

Claim 18. (Original) The processor of claim 17, including logic to determine the width of the result of the operation,

circuitry to store said width and selected bits of said result, and circuitry to store said result.

Claim 19. (New) A multistage microprocessor pipeline structure for executing instructions comprising: an instruction cache, a decoder, a register file, an arithmetic

logic unit, and a cache memory, said register file, arithmetic logic unit (ALU) and cache memory organized as a plurality of slices adapted to be enabled selectively depending on a width of instruction operands and operation results, each slice comprising a reduced bit width portion of said register file, a reduced bit width portion of said arithmetic logic unit, and a reduced bit width portion of the cache memory, wherein all of said slices are enabled to operate in parallel when a full bit width processing operation is executed, or, only a minimum required number of slices are enabled to operate if an operation is determined to be narrower than full bit width, one or more said slices being enabled for operation on a cycle-by-cycle basis during the execution of instructions,

whereby instructions from said instruction cache and decoded for operation by the decoder, and registers used in the operation are detected and register contents are input to the arithmetic logic unit which executes the instruction, said microprocessor pipeline structure further comprising:

a register value information means for receiving from said decoder an identity of registers used in the operation, and outputting information about values included in those registers; and,

a width determination means for receiving outputs from said decoder and said register value information means, and generating one or more width control signals used to enable said one or more slices required for execution of the operation,

wherein only those slices enabled access simultaneously corresponding portions of said register file whose contents are needed for execution of the operation; said contents being input in parallel to the portions of the arithmetic logic unit which execute the operation, and said ALU results generated are written in parallel to said portions of said register file selected to receive those results.

Claim 20. (New) The multistage microprocessor pipeline structure as claimed in Claim 19, wherein if an operation accesses said cache memory, only those portions of the cache memory that participate in the operation are accessed in parallel.

Claim 21. (New) The multistage microprocessor pipeline structure as claimed in Claim 20, wherein a data carry operation proceeds from a lesser significant slice to a more significant slice.